Technical Memorandum No. 159

SACLANT ASW
RESEARCH CENTRE

NUMERICAL SOLUTION OF A SYSTEM OF LANCHESTER DIFFERENTIAL EQUATIONS

by

SVEIN HAUGBERG

1 DECEMBER 1970

NATO

VIALE SAN BARTOLOMEO, 400
I-19026-LA SPEZIA, ITALY

TECHNICAL MEMORANDUM NO. 159

SACLANT ASW RESEARCH CENTRE
Viale San Bartolomeo 400
I 19026 - La Spezia, Italy

NUMERICAL SOLUTION OF A SYSTEM OF LANCHESTER DIFFERENTIAL EQUATIONS

by

Svein B. Haugberg

1 December 1970

APPROVED FOR DISTRIBUTION

[illegible]

Ir M.W. van Batenburg
Director

Manuscript Completed:
21 September 1970

TABLE OF CONTENTS

	Page
ABSTRACT	1
INTRODUCTION	2
1. THE RUNGE-KUTTA-MERSON METHOD OF SOLUTION	4
2. COMPUTER PROGRAM	7
2.1 Introduction	7
2.2 The Main Program	7
2.3 The Procedures MERSON and FUNC	8
2.4 Output and Input	10
2.5 Limitations, Timing	14
3. SAMPLE PROBLEMS	16
3.1 Problem 1. Lanchester Linear Law with Replacement	16
3.2 Problem 2: Lanchester Mixed Law	21
3.3 General Remarks about the Input Functions	27
REFERENCES	28
APPENDIX A COMPUTER PROGRAM LISTINGS	29

List of Figures

1. Flow diagram for the procedure Merson	9
2. Number of units vs time - Problem 1	19
3. Numbers of Unit 1 vs numbers of Unit 2	19
4. Loss rate vs time	20
5. Introduction rate vs time	20
6. Number of units vs time - Problem 2	24
7. Numbers of Unit 1 vs numbers of Unit 2	24
8. Loss rate vs time	25
9. Percentages of loss rate for Unit 1	25
10. Percentages of loss rate for Unit 2	26
11. Exchange rate	26

NUMERICAL SOLUTION OF A SYSTEM OF LANCHESTER DIFFERENTIAL EQUATIONS

by

Svein D. Haugberg

ABSTRACT

Using the Runge-Kutta-Merson method a computer program was developed for the numerical solution of a system of generalized Lanchester equations. A detailed description is given of the input-output features of the computer program and of the algorithm used to solve the equations. Two sample problems illustrate the use of the program and its flexibility.

INTRODUCTION

The purpose of this memorandum is to describe the development of a computer program for numerically solving a system of generalized Lanchester equations.

The system of generalized Lanchester differential equations [Ref. 1] may be written as follows:

$$\dot{\underline{X}} = -\underline{f}(\underline{X}) + \underline{r} , \qquad \text{[Eq. 1]}$$

where $\underline{f}(\underline{X}) \geq 0$ for $\underline{X} \geq 0$, and with known initial values $X(0)$.

The vector $\underline{X} = (X_1, X_2, \ldots, X_n)$, $X_i = X_i(t)$, describes the number of units engaged in combat in each of n different classes as a function of time. The components $f_i(\underline{X})$ of $\underline{f}(\underline{X})$ are the expected number of units of class i lost per unit time. The vector $\underline{r} = (r_1, r_2, \ldots, r_n)$, $r_i = r_i(t)$, is the rate of introduction of new units in each of the n classes. Each $f_i(\underline{X})$ is assumed to be the sum of m_i contributions:

$$f_i(\underline{X}) = \sum_{j=1}^{m_i} f_{ij}(\underline{X}) , \qquad \text{[Eq. 2]}$$

where $f_{ij}(\underline{X})$ is the j^{th} contribution to $f_i(X)$. These contributions are generally associated with different factors causing attrition of type i units.

The remainder of this memorandum proceeds along the following lines. First, the method of numerically solving the system of Lanchester equations, called the Runge-Kutta-Merson Method, is discussed. Second, a computer program that performs the numerical solution is described. Finally, two sample problems and their computer solutions are presented. A complete listing of the computer program appears in Appendix A.

1. THE RUNGE-KUTTA-MERSON METHOD OF SOLUTION

The Runge-Kutta method of numerical integration is one of several fourth-order processes and is applicable to all systems of 1st order differential equations of the form $\underline{\dot{X}}(t) = \underline{g}[\underline{X}(t), t]$, and therefore also to all systems of higher order that can be reduced to the same form. To start the integration, initial values of t_0, $\underline{X}(t_0)$ are required. The method employs a constant step length in the integration. A difficulty with this method is that it is often not clear how to determine a proper step size to achieve a known local truncation error at each step. The method of Merson overcomes this difficulty by automatically determining the step length required to obtain a predetermined accuracy.

If we define $\underline{g}(\underline{X}, t)$ to be

$$\underline{g}(\underline{X}, t) = -\underline{f}(\underline{X}) + r, \qquad \text{[Eq. 3]}$$

then the system of Lanchester equations described above becomes

$$\underline{\dot{X}} = \underline{g}(\underline{X}, t). \qquad \text{[Eq. 4]}$$

This more compact form will be used for the following discussion of the Runge-Kutta-Merson method.

Given t_0, an initial time, $\underline{X}(t_0)$, the corresponding initial value of $\underline{X}$, and h, the step length, the Runge-Kutta-Merson method [Ref. 2] allows one to integrate $\underline{g}(\underline{X}, t)$ to a later time $t_1 = t_0 + h$ and obtain $\underline{X}(t_1)$ by the following procedure.

First, let t_a, t_b, t_c and t_d be four intermediate points in one integration step h from t_0 to t_1 selected as follows:

$$t_a = t_b = t_o + \frac{h}{3}$$

$$t_c = t_o + \frac{h}{2} \qquad [\text{Eq. } 5]$$

$$t_d = t_o + h = t_1$$

Second, define $X_i^j = X_i(t_j)$ and $g_i^j = g_i(\underline{X}(t_j),\ t_j)$ for $j = o, a, b, c, d, 1$.

Third, compute $\underline{X}(t_1)$ by performing the following five steps (substeps of the integration step h) for each value of i:

$$X_i^a = X_i^o + \frac{h}{3} g_i^o$$

$$X_i^b = X_i^o + \frac{h}{6} (g_i^o + g_i^a)$$

$$X_i^c = X_i^o + \frac{h}{8} (g_i^o + 3 g_i^b) \qquad [\text{Eq. } 6]$$

$$X_i^d = X_i^o + \frac{h}{2} (g_i^o + 3g_i^b + 4g_i^c)$$

$$X_i^1 = X_i(t_1) = X_i^o + \frac{h}{6} (g_i^o + 4g_i^c + g_i^d)$$

The method is of fourth order in h.

Merson has shown that if the step length h is small enough, so that we can represent $g_i[\underline{X}(t),\ t]$ by a linear approximation in $\underline{X}$ and t, a good estimation of the truncation error in the computed X_i^1 is $\frac{1}{5}\ |X_i^d - X_i^1|$. The relative error then becomes

$$q_i = \frac{h}{5\,X_i^1} \left|\frac{1}{3}\, g_i^o - \frac{3}{2}\, g_i^b + \frac{4}{3}\, g_i^c - \frac{1}{6}\, g_i^d\right| \qquad [\text{Eq. } 7]$$

The method of Merson for automatically adjusting the step length h is as follows.

Given a predetermined relative accuracy e, if $q_i > e$ for at least one i, the step length will be halved and the integration repeated with this new h. This will go on until $q_i < e$ for all i. If, on the other hand, $q_i < e/32$ for all i, the step length will be doubled, and this new h will be used in the next step.

To continue the integration, let t_1 and $\underline{X}(t_1)$ be the new initial values t_0 and $\underline{X}(t_0)$ respectively, and repeat the procedure described above.

The effect of non-linearities in $\underline{g}$ may overestimate the smallness of the step length necessary to achieve the required accuracy. It is also possible that the Merson process might underestimate the error, but according to Ref. 2 no example of this type of behaviour has been encountered.

Applied to a computer program, the advantage of the Runge-Kutta-Merson method consequently is that the integration always will be performed with the predetermined accuracy and that the step length is never much smaller than necessary, which saves computer time.

2. COMPUTER PROGRAM

2.1 Introduction

A computer program that performs numerical solutions of the system of generalized Lanchester equations described in the previous section has been written in ALGOL for the Elliott 503 computer. The results produced by the program are the values of $\underline{X}$ and the other time-dependent variables in Eqs. 1 and 2, for certain discrete values of time in a specified region $[t_{in}, t_m]$, where t_{in} is the initial time, $\underline{X}(t_{in})$ is assumed known. The program is built around a procedure MERSON, based on the Runge-Kutta-Merson method. For given values of $\underline{X}$ at a time t, the procedure MERSON computes $\underline{X}$ at a later time $t + Dt$ by successively increasing time with variable step length h until $t + Dt$ is reached.

Another procedure, FUNC, contains all the functions $f_{ij}(\underline{X})$ and $r_i(t)$, and this procedure computes the values of the derivatives $\dot{\underline{X}}$ when called by MERSON.

2.2 The Main Program

The central part of the main program consists of a loop that starts with the initial values t_{in} and $\underline{X}(t_{in})$ and computes $\underline{X}$ at a later time $t = t_{in} + Dt$ by calling MERSON. The values of $\underline{X}$ at this new time are used to compute the other time-dependent variables for this t. These are obtained by calling FUNC. The values of all the time-dependent variables are stored in arrays for later output. In the next run through the loop the last values of t and $\underline{X}$ are new initial values, and MERSON computes $\underline{X}$ at $t + Dt$. The loop terminates at $t = t_m$ ($t_m - t_{in}$ is an integer multiple of Dt).

The rest of the main program provides for output. Some flexibility exists in the choice of variables output; the different possibilities will be given in the description of input and output.

2.3 The Procedures MERSON and FUNC

The Runge-Kutta-Merson method is described in Chapter 1. This method is well suited for implementation on an electronic computer [Ref. 3]. Equations 5, 6 and 7, together with the accuracy test, constitute the main part of the procedure MERSON. The following quantities are required by the procedure MERSON when it is called in the main program to compute $\underline{X}(t + Dt)$ for known $\underline{X}(t)$:

n, number of equations.

t, start point of integration.

$z = t + Dt$, end point of integration.

$x_i(t)$, $(i = 1, n)$, the values of $\underline{X}$ in the start point of integration.

h, a starting value for the step length.

h min, lower bound for the step length. (This prevents indefinite cycling, otherwise possible in some cases.)

e, desired bound for relative accuracy.

With these quantities given, the procedure MERSON performs one step h according to Eqs. 6 and 7 and the accuracy test with $t_0 = t$, and calculates $\underline{X}(t_1)$. The derivatives $\dot{x}_i = g_i$ $(i = 1, n)$, needed in every substep, are calculated by calling FUNC before each of them. The computations are continued by letting t_1 and $\underline{X}(t_1)$ be the new t_0 and $\underline{X}(t_0)$ respectively. When the integration has reached a stage where $t_0 + h$ exceeds the end point of integration, z, a final step length $z - t_0$ will be used to reach z exactly. For this last special step a better accuracy than the one needed will result because $z - t_0 < h$.

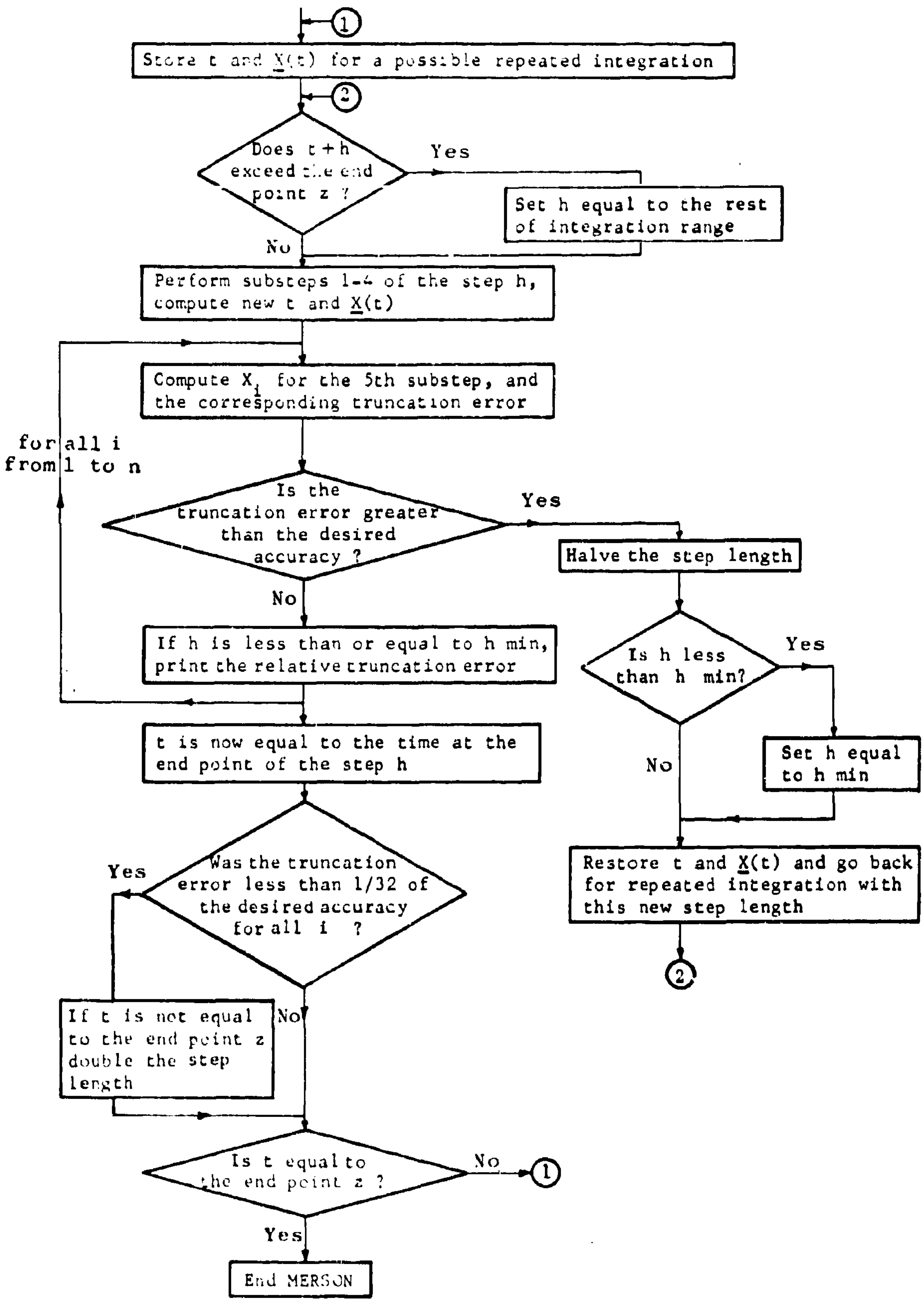
1
Store t and $\underline{X}(t)$ for a possible repeated integration
2
Does t + h exceed the end point z ?
Yes
Set h equal to the rest of integration range
No
Perform substeps 1-4 of the step h, compute new t and $\underline{X}(t)$
Compute X_i for the 5th substep, and the corresponding truncation error
for all i from 1 to n
Is the truncation error greater than the desired accuracy ?
Yes
Halve the step length
No
If h is less than or equal to h min, print the relative truncation error
Is h less than h min?
Yes
Set h equal to h min
No
t is now equal to the time at the end point of the step h
Restore t and $\underline{X}(t)$ and go back for repeated integration with this new step length
2
Was the truncation error less than 1/32 of the desired accuracy for all i ?
Yes
If t is not equal to the end point z double the step length
No
Is t equal to the end point z ?
No
1
Yes
End MERSON

FIG 1 FLOW DIAGRAM FOR THE PROCEDURE MERSON

Results produced by MERSON are:

> t, its value is now that of $z = t + Dt$.
>
> $X_i(t)$, $(i = 1, n)$, the values of $\underline{X}$ at the end point.
>
> $g_i(\underline{X}, t)$, $(i = 1, n)$, the values of the derivatives $\dot{X}_i$
> $\dot{X}_i$ at the end point.
>
> h, adjusted step length.

A flow diagram for the procedure MERSON is shown in Fig. 1.

It appears from the flow diagram that if h becomes less than h min, h min is used as the step length instead of h. This is in order to avoid particularly long run time on the computer. In this case, however, it might happen that the desired accuracy is not satisfied. As an indication for the user of the program, the character followed by the relative truncation error is printed in the table of $X_i(t)$ every time h min is used as the step length.

2.4 Output and Input

The results of the computations are printed out for certain discrete values of t, starting with t_{in} and increasing repeatedly with a constant time interval Dt until t_m is reached. Since the integration of each interval Dt is an independent computation with new initial conditions, these initial conditions can be changed at the start point of any interval. Therefore at these points the analytical form of the functions $f_{ij}(\underline{X})$ and $r_i(t)$ can be changed, and instantaneous introductions of units of any class i can be made. Whenever such changes or introductions occur, the region of integration has to be divided into smaller regions such that all changes and introductions occur at the start point of a region, and that every region is an integral number Dt.

Output

As mentioned above, the program allows for a choice in the selection of variables and combinations of them given as output. The output consists of:

a. The number of units introduced instantaneously and the corresponding instants in time.

b. X_i vs t for $i = 1, n$.

c. X_i vs X_j for specified index pairs (i, j).

d. f_i vs t for specified indices i.

e. r_i vs t for specified indices i.

f. f_{ij}/f_i (in percent) vs t for specified indices i, and specified indices j for each i.

g. f_i/f_j vs t for specified index pairs (i, j).

The last five outputs are optional and may be skipped by not specifying any indices. This output form enables the user of the program to study the importance of instantaneous introductions and changes in attrition functions and introduction rates, and the relative importance of the different contributions $f_{ij}(\underline{X})$ causing attrition of type i units. This will be illustrated by the sample problems in the next chapter.

There are two forms of output: paper tape and plotted. The paper tape output is punched on the Number 1 punch, and, of course, must be run through the Flexowriter to obtain a printer output. The plotted output is generated by the Calcomp plotter.

Input

There are two kinds of input to this program: input parameters and input functions. These are listed in Tables 1 and 2.

TABLE 1
LIST OF INPUT PARAMETERS

In the following list the letter in parentheses following the explanation indicates whether the variable is integer (I) or real (R).

N,	number of differential equations	(I)
DT,	the constant time interval between two following output points	(R)
H,	a start value for the step length	(R)
HMIN,	lower bound for the step length	(R)
E,	desired bound for relative accuracy	(R)
MI,	maximum number of contributions m_i of $f_{ij}(\underline{X})$ to $f_i(\underline{X})$ $(i = 1,n)$. $[MI = \max(m_i), i = 1,n]$.	(I)
NI,	number of time regions according to the instantaneous introductions and the change of functions	(I)
NS,	number of pairs (X_i, X_j) to be plotted (X_i vs X_j, t as a parameter)	(I)
N1,	number of outputs (print and plot) of f_i vs t	
NR,	number of outputs (plot) of r_i vs t	(I)
N2,	number of outputs (print and plot) of f_{ij}/f_i vs t (number of different indices i)	(I)
NP,	number of outputs (print and plot) of f_i/f_j vs t	(I)
MA[L] (L = 1, NI),	number of Dt in each of NI regions (the lengths of the regions are MA[L] * DT).	(I)
II,	initial time	
DX[L,J] (J = 1, N for each L = 1, NI),	instantaneous introductions of units in the start point of region L. The initial values of X_i $(i = 1, n)$ are included here (L = 1).	(R)

U, V (K = 1, NS), specification of indices in NS plots of X_i vs X_j (I)

U (I = 1, N1), specification of index in N1 outputs of f_i vs t (I)

U, M1 (J = 1, N2), specification of index i and number of different indicex j for each i in outputs of f_{ij}/f_i vs t (I)

V (K = 1, M1 for each J = 1, N2), specification of index j in M1 outputs of f_{ij}/f_i vs t (I)

U, V (J = 1, NP), specification of indices i and j in NP outputs of f_i/f_j vs t (I)

U (I = 1, NR), specification of index in NR outputs of r_i vs t (I)

The input data must be given numerical values and punched on a data tape in the following order:

N, DT, H, HMIN, E, MI, NI, NS, N1, NR, N2, NP

MA[L] for L = 1, NI

TI

(DX[L, J] for J = 1, N) for L = 1, NI

(U, V) for K = 1, NS

U for I = 1, N1

(U, M1, (V for K = 1, M1)) for J = 1, N2

(U, V) for J = 1, NP

U for I = 1, NR

TABLE 2

LIST OF VARIABLES USED IN THE INPUT FUNCTIONS

T,	corresponds to t
X[I] (I = 1, N),	corresponds to $X_i(t)$
FX[I, J] (J = 1, M1 for each I = 1, N),	corresponds to $f_{ij}(\underline{X})$
RT[I] (J = 1, N),	corresponds to $r_i(t)$

In contrast to the input parameters, which are entered as numerical values, the input functions $f_{ij}(\underline{X})$ and $r_i(t)$ are entered in the form of program statements in the procedure FUNC. How this is done is explained in Chap. 3.

2.5 Limitations, Timing

Because of the limited capacity of the computer store, the upper bound for the number of differential equations the program can solve is 19. However the practical bound is lower because the running time for 19 equations is probably excessive. The limitations resulting from the output format are:

Maximum number of equations: 14
Maximum initial value of X_i: 9999 units
Maximum length of the region of integration: 200 days.

These can be changed if necessary by altering the program.

It is very difficult to estimate the running time for a given set of equations. The running time depends on the number of equations, the desired bound for relative accuracy, the form of the equations, the length of the region of integration and the output form. However, the following results give some indication of the running time.

For the two sample problems in Chapter 3 the number of equations was 2, the desired bound for relative accuracy was 0.01, and the length of the region of integration was 100 days. The first has one contribution to each attrition function $f_i(\underline{X})$, and the second has two contributions. The running times were 5 and 11 minutes respectively.

A system consisting of 5 equations and a maximum of two contributions to an attrition function has also been run. The running-time was approximately 30 min.

3. SAMPLE PROBLEMS

Two sample problems have been solved to illustrate the possibilities of the program. The first one shows how to make instantaneous introductions of units and how to change the analytical form of the introduction rates. The second one shows how to manipulate a system with more than one contribution to the attrition of some classes of units.

3.1 Problem 1: Lanchester Linear Law with Replacement

Consider the system of two simultaneous differential equations

$$\dot{X}_1 = -\alpha X_1 X_2 + r_1$$
$$\dot{X}_2 = -\beta X_1 X_2 + r_2$$

with initial values $X_1(0) = 1000$, $X_2(0) = 50$, and attrition coefficients $\alpha = 5 \cdot 10^{-4}$, $\beta = 2 \cdot 10^{-5}$. This is a system of Lanchester equations with

$$f_1(\underline{X}) = f_{11}(\underline{X}) = \alpha X_1 X_2$$
$$f_2(\underline{X}) = f_{12}(\underline{X}) = \beta X_1 X_2$$

[only one contribution $f_{ij}(\underline{X})$ to each $f_i(\underline{X})$].

An additional 400 units of class 1 are introduced at time $t = 20$.

Introduction rates:

$$r_1(t) = 0 \text{ for all } t$$
$$r_2(t) = \begin{matrix} 0 \text{ for } t \leq 50 \\ 0.1 \text{ for } t > 50 \end{matrix}$$

The following part of the optional output is desired:

a. X_2 vs X_1 (t as a parameter)

b. r_i vs t, i = 1, 2

c. r_2 vs t

Input parameters. Let the region of integration be 100 days. According to the instantaneous introduction and the change of introduction rate, the region of integration is divided into three sub-regions: 0-20, 20-50, and 50-100. The instantaneous introductions must be specified at the start point of each sub-region. This means that the following six numbers must be given as input:

	Class 1	Class 2
t = 0	1000	50
t =20	400	0
t =50	0	0

The time interval Dt must be chosen such that the three sub-regions become an integer multiple of Dt. Let Dt = 5. This implies that the number of time increments Dt in each sub-region is 4, 6 and 10 respectively. Furthermore, let the initial value for the step length be 1 day, the lower bound for the step length be 0.01, and the desired bound for the relative accuracy be 0.01 (1%). This gives the following scheme of input data to be punched on paper tape (for definition of input parameters see Table 1):

```
DATA:

2  5  1  0.01  0.01  1  3  1  2  1  0  0
4  6  10
0
1000  50
 400   0
   0   0
2  1
1  2
2

END OF DATA;
```

Input functions. The input functions $f_{ij}(\underline{X})$ and $r_i(t)$ are easily inserted into the procedure FUNC by copying the existing paper tape for the procedure up to and including the line

"comment Insert the input functions;"

and punching the following four statements:

```
FX[1,1]:=0.0005*X[1]*X[2] ;
FX[2,1]:=0.00002*X[1]*X[2];
RT[1]:=0.0;
if L=1 or L=2 then RT[2]:=0.0 else RT[2]:=0.1;
```

(The three sub-regions are identified by the values L = 1, L = 2 and L = 3 respectively.) Then copy the rest of the tape. The procedure FUNC then takes the following form:

```
£Numerical'solution of a system of Lanchester differential
 equations?;

begin procedure FUNC(N,MI,T,X,F,FXS,FX,RT,L);
   value N,MI;real T;integer N,MI,L;
   array X,F,FXS,FX,RT;
   begin integer J,K;
      comment Insert the input functions;

      FX[1,1]:=0.0005*X[1]*X[2];
      FX[2,1]:=0.00002*X[1]*X[2];
      RT[1]:=0.0;
      if L=1 or L=2 then RT[2]:=0.0 else RT[2]:=0.1;

      for J:=1 step 1 until N do
      begin FXS[J]:=0.0;
         for K:=1 step 1 until MI do
         FXS[J]:FXS[J]+FX[J,K];
         F[J]:=-FXS[J]+RT[J];
      end;
   end FUNC;
```

Output. The numerical values of all the output variables are given in printed tables obtained from the output paper tape. The plotted output is presented in Figs. 2 to 5.

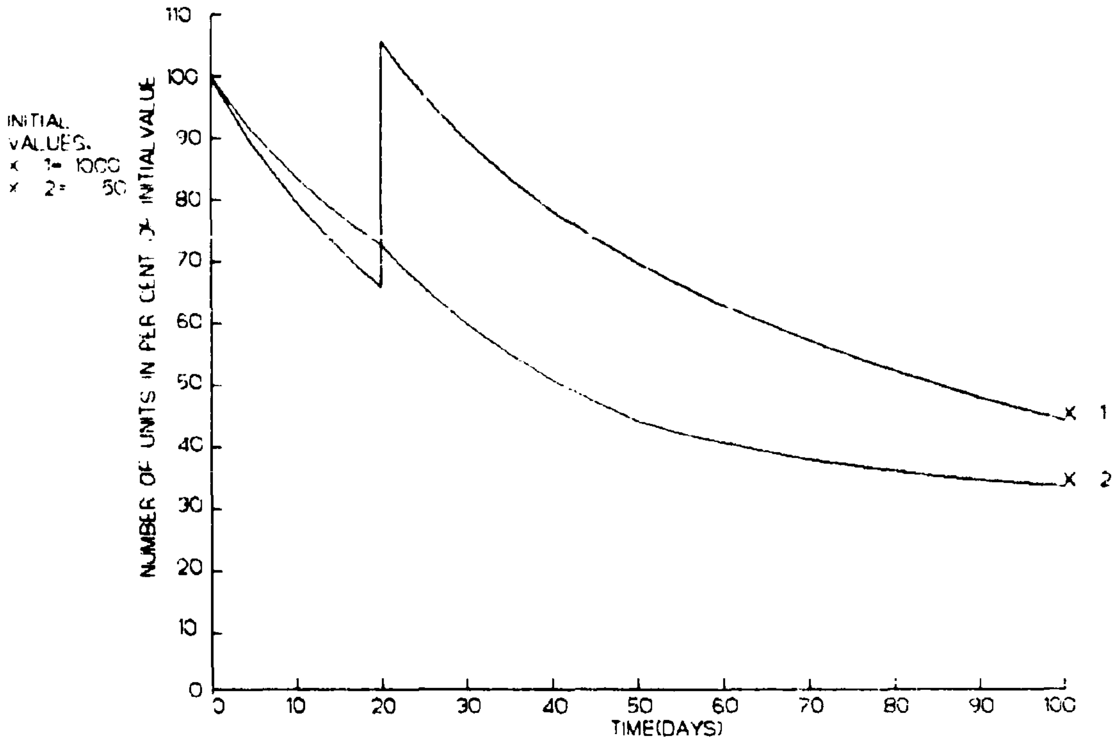

FIG. 2 NUMBER OF UNITS vs TIME

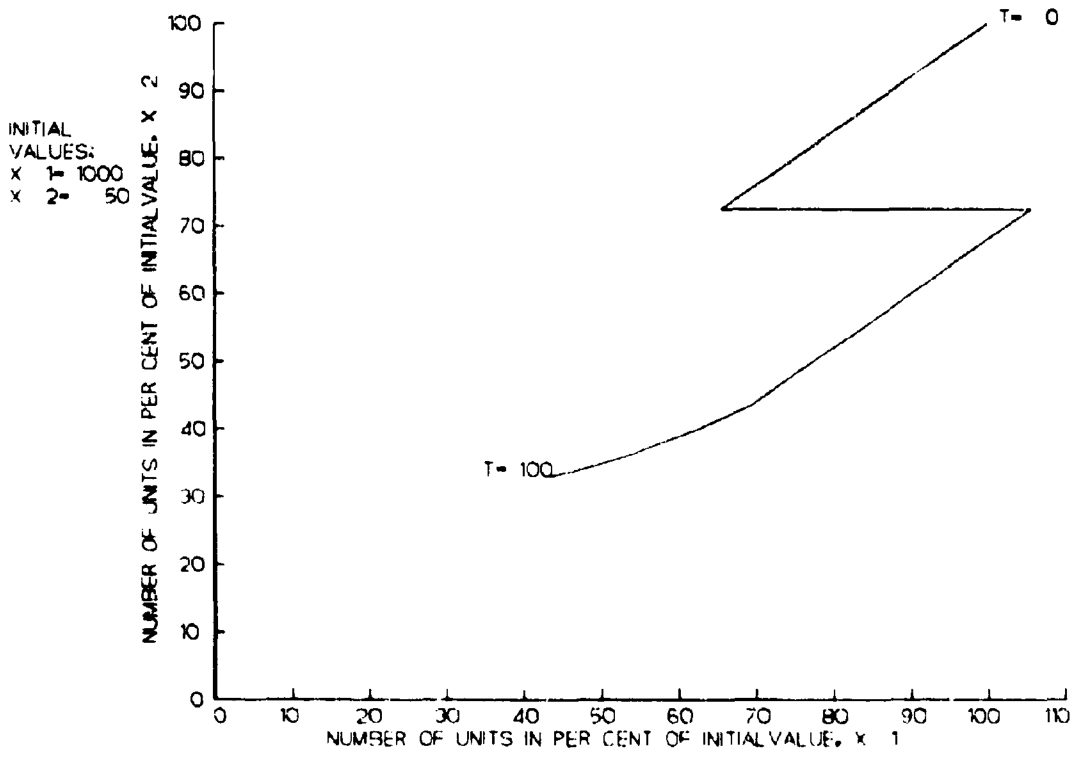

FIG. 3 NUMBERS OF UNIT 1 vs NUMBERS OF UNIT 2

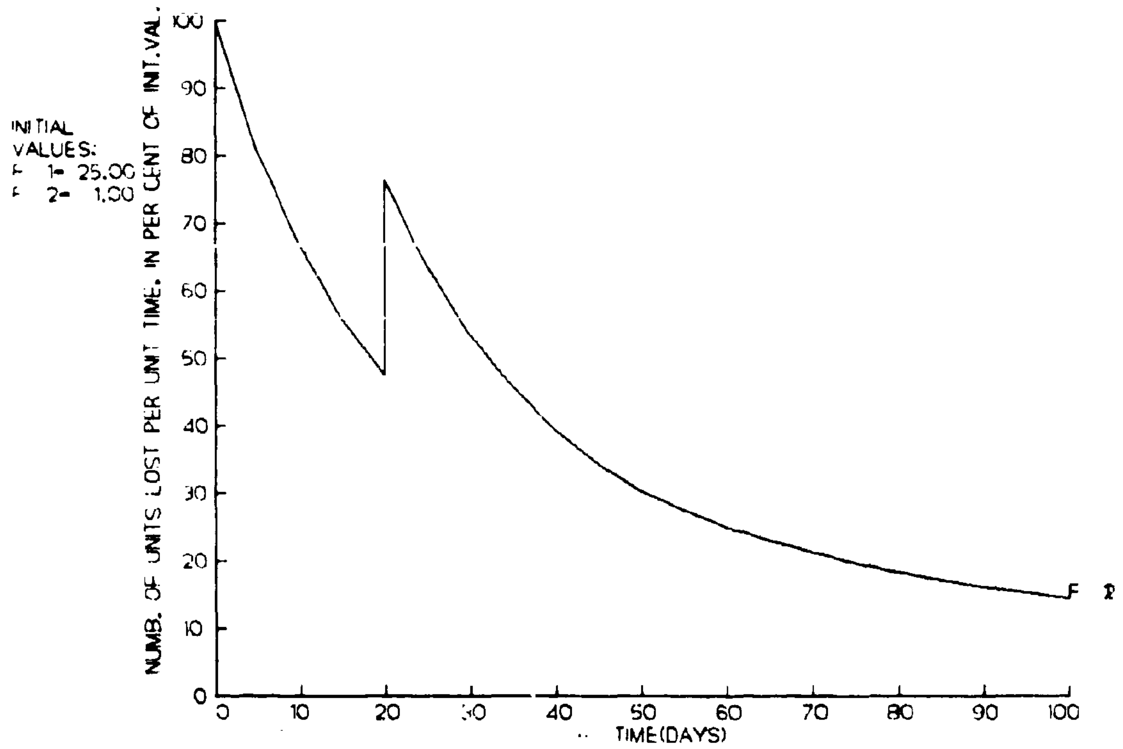

FIG. 4 LOSS RATE vs TIME

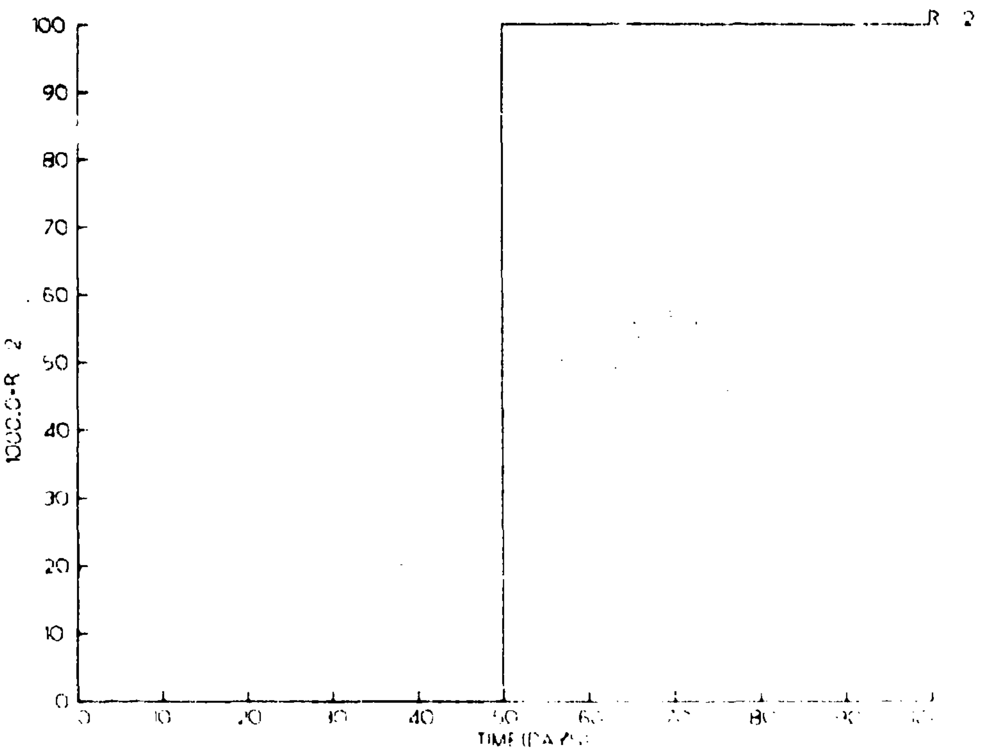

FIG. 5 INTRODUCTION RATE vs TIME

Figure 2 shows the number of units in class 1 and 2 as a function of time. The effect of the instantaneous introduction of units in class 1 at $t = 20$, and the change to a non-zero introduction rate in class 2 at $t = 50$ is obvious. However, the result of the combat, if continued. does not become clear from this figure. The plotting of X_2 vs X_1 with time as a parameter in Fig. 3 is very useful in examining this. The instantaneous introduction changes an apparent victory for class 2 to an apparent victory for class 1. However, after changing the introduction rate of class 2 from zero to a constant non-zero value, it seems as if class 2 will win the combat.

Figure 4 shows the number of units lost per day as a function of time for class 1 and 2. The two curves are identical in this case because f_1 is proportional to f_2 and the plotted values are given as a percentage of the initial values.

Figure 5 shows the introduction rate of class 2 as a function of time.

3.2 Problem 2: Lanchester Mixed Law

Consider the system of two simultaneous Lanchester differential equations

$$\dot{X}_1 = -(\alpha_1 X_1 X_2 + \alpha_2 X_2) + r_1$$

$$\dot{X}_2 = -(\beta_1 X_1 X_2 + \beta_2 X_2) + r_2$$

with initial values $X_1(0) = 1000$, $X_2(0) = 50$, and attrition coefficients $\alpha_1 = 4.10^{-4}$, $\alpha_2 = 0.1$, $\beta_1 = 1.8 \cdot 10^{-5}$, $\beta_2 = 0.002$.

This system has two contributions, $f_{ij}(\underline{X})$, to each $f_i(\underline{X})$:

$$f_1(\underline{X}) = f_{11}(\underline{X}) + f_{12}(\underline{X})$$

$$f_2(\underline{X}) = f_{21}(\underline{X}) + f_{22}(\underline{X})$$

where

$$f_{11}(\underline{X}) = \alpha_1 X_1 X_2, \quad f_{12}(\underline{X}) = \alpha_2 X_2,$$

$$f_{21}(\underline{X}) = \beta_1 X_1 X_2, \quad f_{22}(\underline{X}) = \beta_2 X_2.$$

Let the introduction rate $r_1(t)$ be zero for all t.

The following part of the optional output is desired:

a. X_2 vs X_1 (t as a parameter)

b. f_i vs t, $i = 1, 2$.

c. f_{ij}/f_i vs t, $j = 1, 2$ for each $i = 1, 2$.

d. f_1/f_2 vs t.

Input parameters. Let the length of the region of integration, the interval Dt, the initial value for the step length, the lower bound for the step length, and the desired bound for the relative accuracy be the same as in Problem 1. Since neither instantaneous introductions nor analytical changes of the functions $f_{ij}(\underline{X})$ or $r_i(t)$ occur, there will be no division of the region of integration. This leads to the following scheme of input data to be punched on paper tape (for definition of input parameters see Table 1):

```
DATA:

2  5  1  0.01  0.01  2  1  1  2  0  2  1
20
0
1000  50
2  1
1  2
1  2  1  2
2  2  1  2
1  2

END OF DATA;
```

Input functions. In the same way as in Problem 1, the existing paper tape for the procedure FUNC is copied and the following statements inserted;

```
FX[1,1]:=0.0004*X[1]*X[2]; FX[1,2]:=0.1*X[2];
FX[2,1]:=0.000018*X[1]*X[2]; FX[2,2]:=0.002*X[2];
RT[1]:=RT[2]:=0.0;
```

Output. The numerical values of all the output variables are given in printed tables obtained from the output paper tape. The plotted output is presented in Figs. 6 to 11.

Figure 6 shows the number of units in class 1 and 2 as a function of time.

Figure 7 contains the plotting of X_2 vs X_1 with time as a parameter. This figure clearly shows the result of the combat, if continued.

Figure 8 shows the number of units lost per day as a function of time for class 1 and 2.

Figures 9 and 10 are very convenient in examining the relative importance of the different contributions to an attrition function $f_i(\underline{X})$. Figure 9 shows that $f_{11}(\underline{X})$ constitutes the greatest part of $f_1(\underline{X})$ in the beginning of the combat. However, as the time increases, $f_{12}(\underline{X})$ increases while $f_{11}(\underline{X})$ decreases; consequently $f_{12}(\underline{X})$ will dominate after some time. The same relation appears in Fig. 10 for $f_{21}(\underline{X})$ and $f_{22}(\underline{X})$.

Figure 11 indicates that number of class 1 units lost per class 2 unit lost increases with time, emphasizing the fact that class 2 will win the combat.

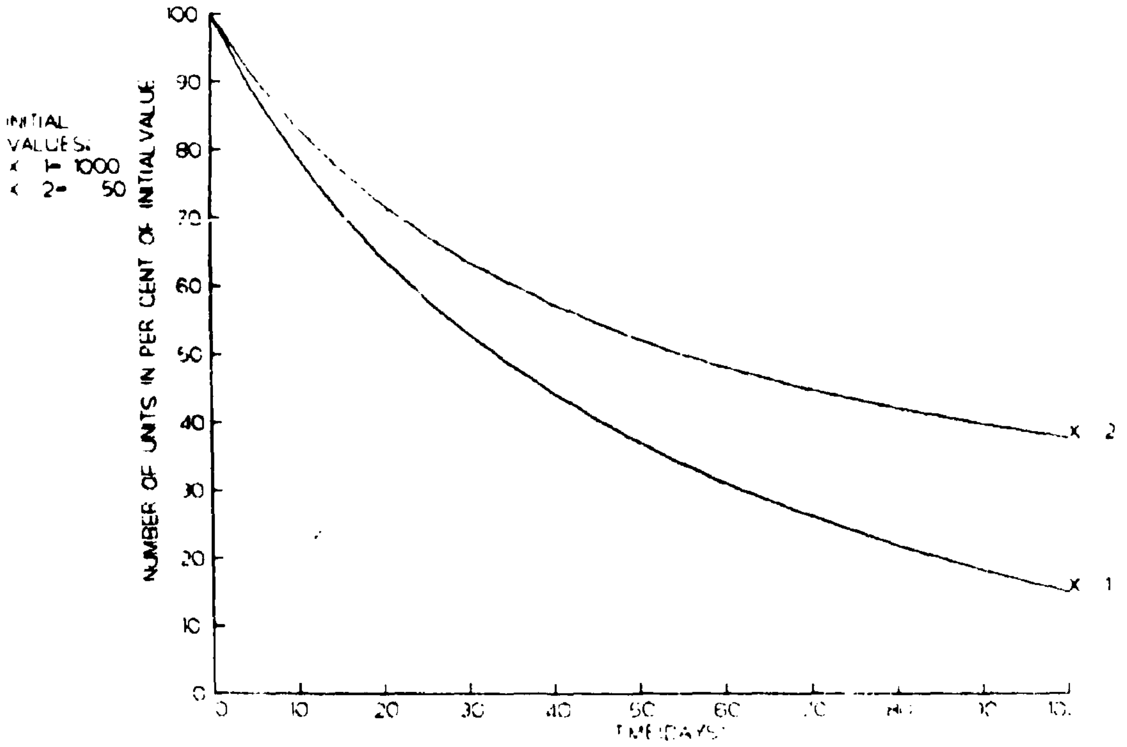

FIG. 6 NUMBER OF UNITS vs TIME

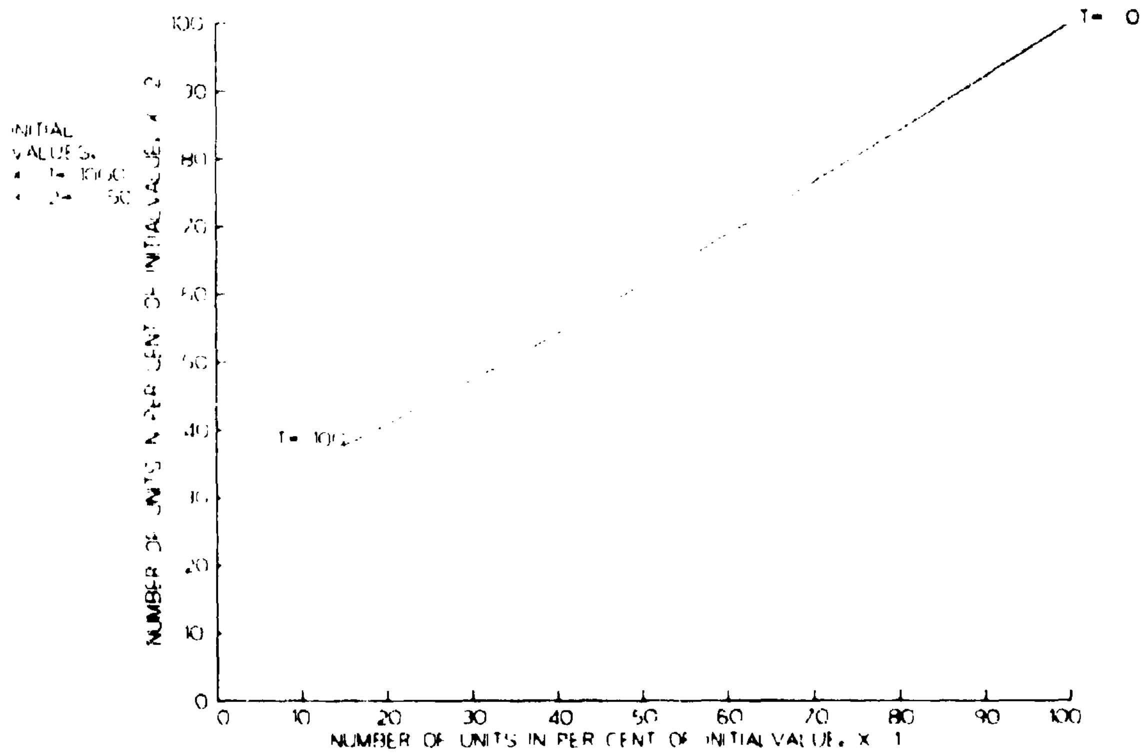

FIG. 7 NUMBERS OF UNIT 1 vs NUMBERS OF UNIT 2

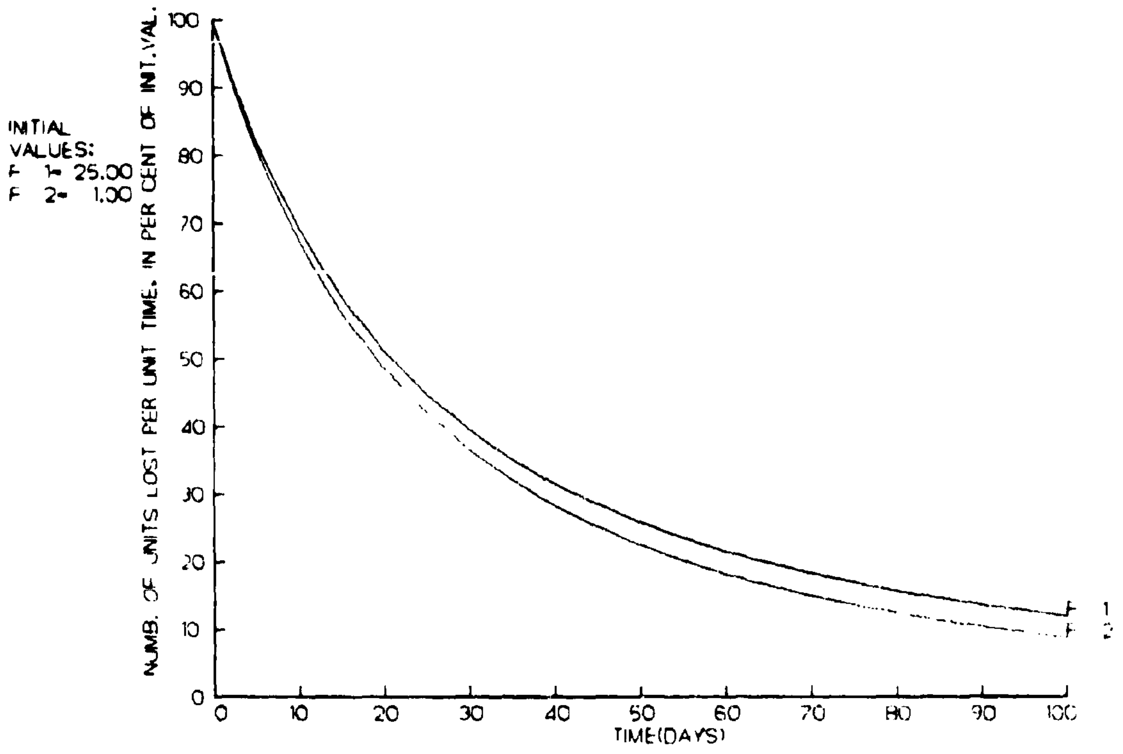

FIG. 8 LOSS RATE vs TIME

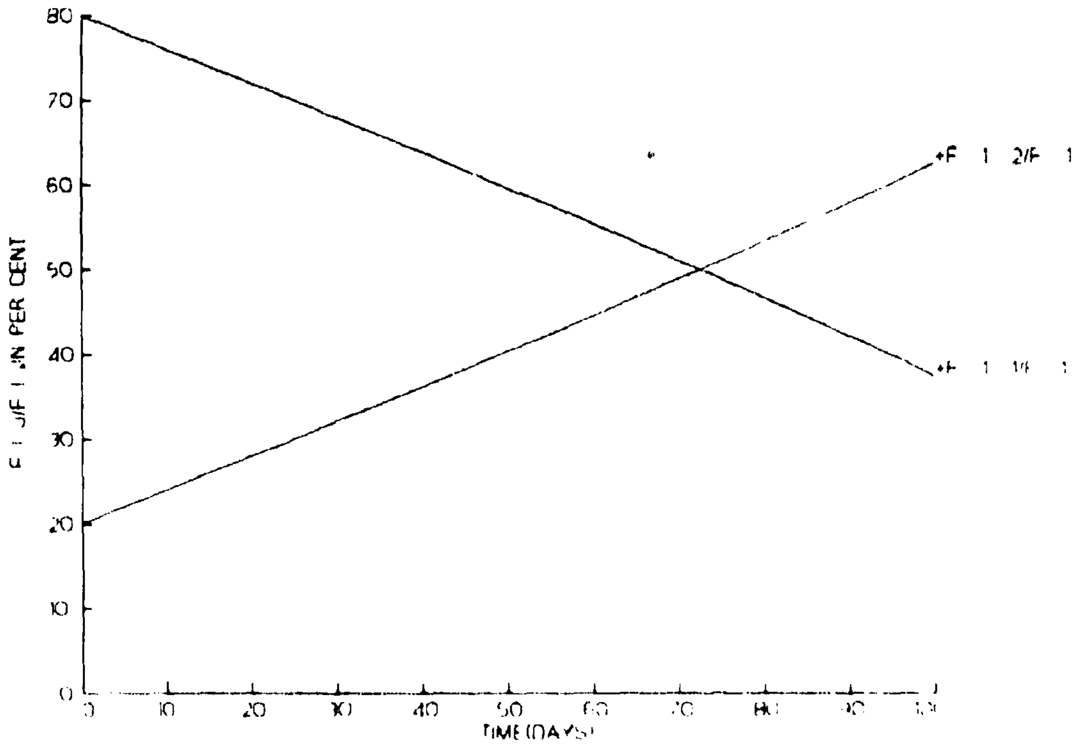

FIG. 9 PERCENTAGES OF LOSS RATE FOR UNIT 1

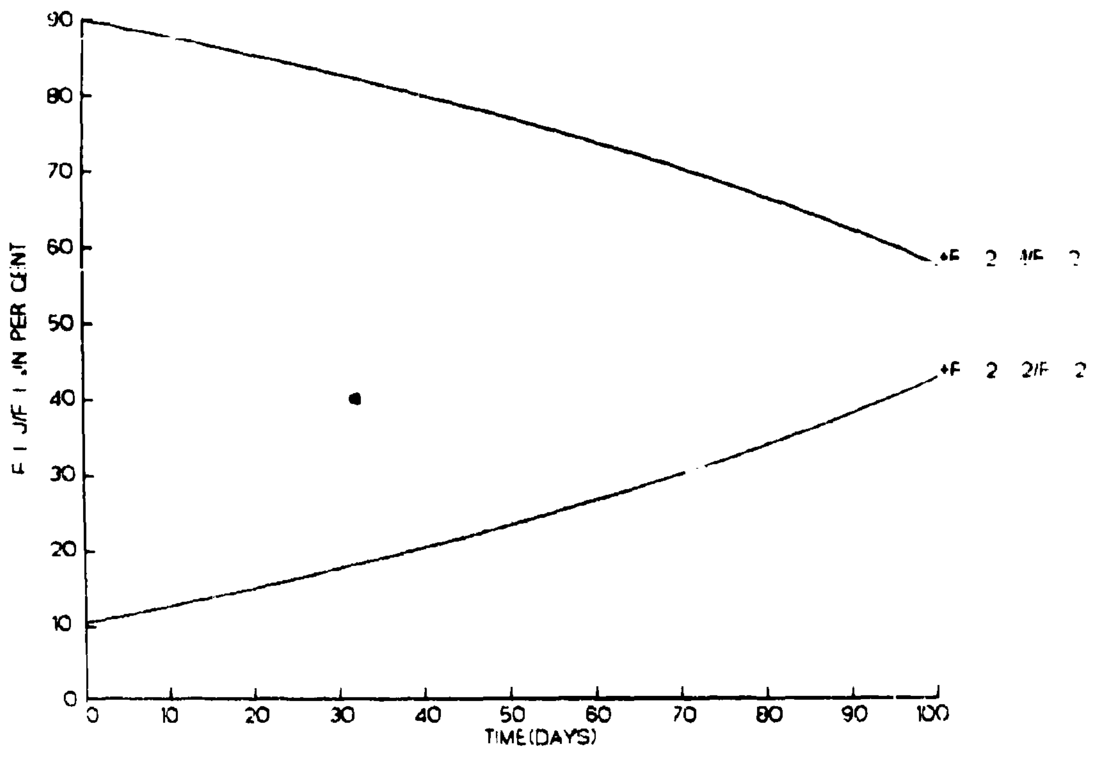

FIG. 10 PERCENTAGES OF LOSS RATE FOR UNIT 2

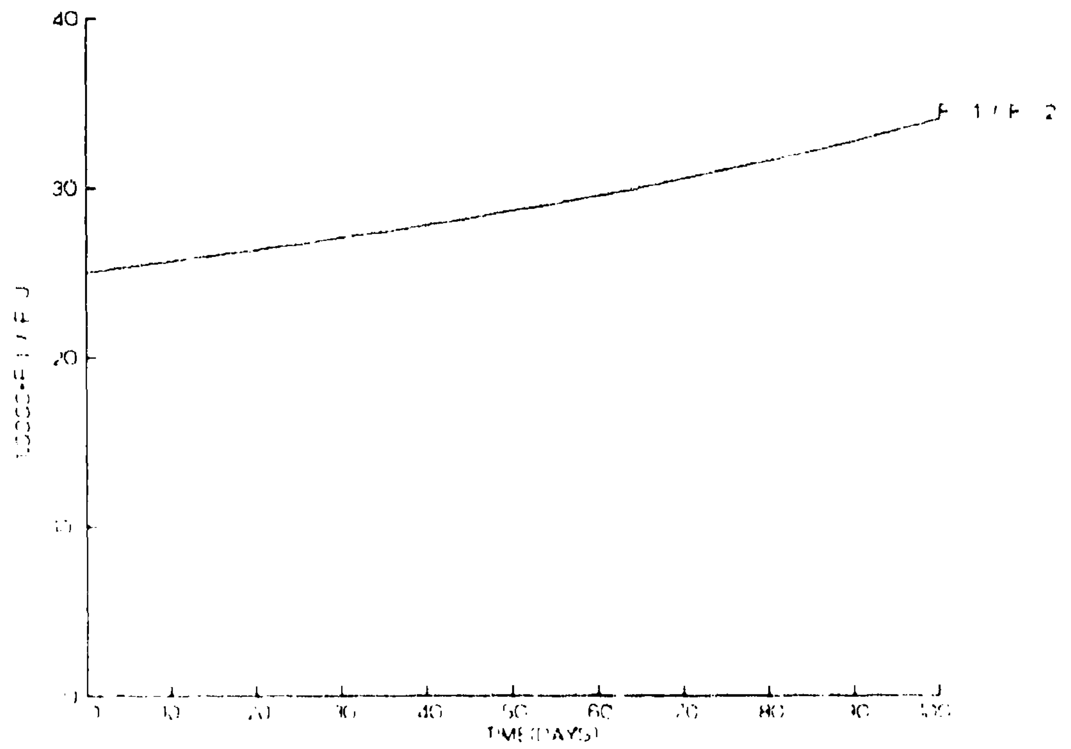

FIG. 11 EXCHANGE RATE

3.3 General Remarks About the Input Functions

Sample problem 1 shows how to change the analytical form of one function $r_i(t)$ for some t. The same is of course possible for any $r_i(t)$ and any $f_{ij}(\underline{X})$. If some change exists for a function, say $f_{21}(\underline{X})$, the analytical form of this function must be specified for every sub-region (L = 1, NI) in a statement of the form indicated in Problem 1.

Since MI is the maximum number of contributions $f_{ij}(\underline{X})$ to $f_i(\underline{X})$ (i = 1, n), some of the functions FX[I, J] may of course be equal to zero. However, all of them have to be inserted into FUNC(J = 1, MI for each I = 1, N). Likewise, all the functions RT[I] (I = 1, N) must be inserted.

If it happens that two or more of the contributions $f_{ij}(\underline{X})$ for the same index i are not desired as output, these may be put together in one function FX[I, J]. This will reduce the running time on the computer.

REFERENCES

1. K.M. Mjelde and R.R.V. Wiederkehr, "The Approximate Solution of a Generalized Set of Deterministic Lanchester Equations", SACLANTCEN Technical Report No. 172, September 1970, NATO RESTRICTED.

2. L. Fox, "Numerical Solution of Ordinary and Partial Differential Equations", London Pergamon Press, 1962, pp.24-25.

3. J. Christiansen, "Numerical Solution of Ordinary Simultaneous Differential Equations of the 1st Order Using a Method for Automatic Step Change", Numerische Mathematik b.14, 1970, pp. 317-324.

APPENDIX A

COMPUTER PROGRAM LISTINGS

```
£Numerical solution of a system of Lanchester differential equations?;

begin procedure FUNC(N,MI,T,X,F,FXS,FX,RT,L);
   comment This procedure contains the input functions.
       Parameters:
       N,  number of equations.
       MI, maximum number of contributions to one attrition function.
       T,  a variable for the time.
       X,  array containing the number of units in each class at time T.
       F,  array containing the values of the derivatives at time T.
       FXS,array containing the values of the attrition functions at time T.
       FX, array containing the values of the different contributions to
              the attrition functions at time T.
       RT, array containing the values of the introduction rate functions
              at time T.
       L,  variable which has the value 1 when T in first sub-region, 2
              when T in second sub-region .......;
   value N,MI;real T;integer N,MI,L;
   array X,F,FXS,FX,RT;
   begin integer J,K;
       comment Insert the input functions;




       for J:=1 step 1 until N do
       begin FXS[J]:=0.0;
          for K:=1 step 1 until MI do
          FXS[J]:=FXS[J]+FX[J,K];
          F[J]:=-FXS[J]+RT[J];
       end;
   end FUNC;
```

```
  procedure MERSON(N,MI,T,Z,H,HMIN,E,X,F,FXS,FX,RT,LL);
      comment This procedure performs the integration from T to Z.
              Parameters:
              Z,  end point of integration,
              H,  starting value for the step length,
              HMIN,lower bound for the step length.
              E,  desired bound for relative accuracy.;
  value N,MI,HMIN,E;
  real T,Z,H,HMIN,E;
  integer N,MI,LL;
  array X,F,FXS,FX,RT;
  comment Variables:
      Q,  1. The rest of the interval of integration.
          2. A variable used in the computation of the five sub-steps.
          3. 5*(Truncation error).
      U,  1. A variable used in the computations of the five sub-steps.
          2. U=abs(X[I]).
      H3=H/3
      HS, stores H for a possible restoring.
      TS, stores T for a possible repeated integration.
      E5=5*E
      I,J,SW,index variables.
      G,  array storing X for a possible repeated integration.
      P,L,arrays for temporary storing in the computations of the five
          sub-steps.
      Boolean variables:
      BC=true if E<1
      BF=false if the conditions for doubling the step length are not
          satisfied.
      BH=false if the step length is halved.
      BR=false if H is equal to the rest of the interval.
      BX=false if H=HMIN;
  begin real Q,U,H3,HS,TS,E5;
      integer I,J,SW;
      array G,P,L[1:N];
      Boolean BC,BE,BH,BR,BX;
      switch SSS:=SAVE,BACK;
      comment Check some input parameters and initialize;
      if HMIN<0 then HMIN:=0.01*abs(H);
      BH:=BR:=BX:=true;
      BC:=E<1;
      E:=abs(E);
      E5:=5*E;
      H:=abs(H);
SAVE: if BC then
      begin TS:=T;
         for J:=1 step 1 until N do
         G[J]:=X[J];
      end;
```

```
BACK: HS:=H;
      Q:=T+H-Z;
      BE:=true;
      comment Test end of integration range;
      if Q>0 then
      begin H:=Z-T;
         BR:=false;
      end
      Next integrate one step;
      H3:=H/3;
      for SW:=1 step 1 until 5 do
      begin switch SS:=SW1,SWEX,SW3,SW4,SWAX;
         FUNC(N,MI,T,X,F,FXS,FX,RT,LL);
         for I:=1 step 1 until N do
         begin switch S:=ST1,ST2,ST3,ST4,ST5;
            switch SSSS:=NEXT;
            Q:=H3*F[I];goto S[SW];
      ST1:  L[I]:=U:=Q;goto NEXT;
      ST2:  U:=0.5*(Q+L[I]);goto NEXT;
      ST3:  P[I]:=U:=3*Q;
            U:=0.375*(U+L[I]);goto NEXT;
      ST4:  L[I]:=U:=L[I]+4*Q;
            U:=1.5*(U-P[I]);goto NEXT;
      ST5:  U:=0.5*(Q+L[I]);
            Q:=abs(2*U-1.5*(Q+P[I]));
      NEXT: X[I]:=G[I]+U;
            if SW=5 then
            begin if BC then
               begin U:=abs(X[I]);
                  U:=if U<_{10}-3 then E5 else E5*U;
                  comment Test adjustment of the step;
                  if Q>U and EX then
                  begin BR:=true;
                     BH:=false;
                     H:=0.5*H;
                     if H<HMIN then
                     begin H:=HMIN;
                        BX:=false;
                     end The step was halved, restore X and T, and go back
                         for repeated integration with this new step;
                     for J:=1 step 1 until N do
                     X[J]:=G[J];
                     T:=TS;goto BACK;
                  end if Q;
                  if Q>0.03125*U then BE:=false;
               end if BC;
               if not BX then
               begin if I=1 then print punch(1),££1s10??;
                  print punch(1),sameline,£*?,scaled(4),E*Q/U;
               end;
            end if SW;
         end for I;
         goto SS[SW];
```

```
    SW1:  T:=T+H3;goto SWEX;
    SW3:  T:=T+0.5*H3;goto SWEX;
    SW4:  T:=T+0.5*H;
    SWAX: H:=H; comment Dummy statement;
SWEX: end for SW;
        if BC then
        begin comment Test a possible doubling of the step;
           if BM and BH and BR then
           begin H:=2*H;
              BX:=true;
           end;
           BH:=true;
        end if BC;
        if BR then goto SAVE;
        H:=HS;
    end MERSON;




    procedure AXPLOT(TM,XM,R1,R2,I1,I2);
    comment This procedure plots axes and axe values for the maximum
        axe values given.
        Parameters:
        TM, maximum abscissa value.
        XM, maximum ordinate value.
        R1,R2,scaling factors.
        I1,I2,number of divitions required on the two axes;
    real TM,XM,R1,R2;
    integer I1,I2;
    begin integer J;
        I1:=(TM+4.0)/10.0;
        R1:=750.0/(10.0*I1);
        I2:=(XM+4.0)/10.0;
        R2:=600.0/(10.0*I2);
        setorigin(200,R1,R2,1);
        axes(10,10,I1,0,I2,0);
        plotter(10,1);
        for J:=0 step 1 until I1 do
        begin movepen(J*10.0-30.0/R1,-20.0/R2);
           print digits(3),J*10;
        end;
        for J:=0 step 1 until I2 do
        begin movepen(-50.0/R1,J*10.0-6.0/R2);
           print digits(3),J*10;
        end;
    end AXPLOT;
```

```
comment The main program contains a loop calling MERSON repeatedly and
    storing the values of the variables at the end points of every
    interval DT. The rest of the main program provides for output.
    Variables:
    DT, the interval length.
    TM, end point of the region of integration.
    XM,YM,maximum axe values for plotting.
    I,J,K,L,index variables.
    M, index variable for the time output points.
    M1,N2,U,V,integer variables with different applications.
    NI, number of sub-regions of integration.
    NS,N1,NR,N2,NP,inputparameters which provide for the optional output.
    MA, array containing the number of DT in each sub-region.
    MM, array containing the value  of M at the start point and end
        point of every sub-region.
    NN,N3,N4,NM,N5,N6,arrays storing the different indices used in the
        optional output.
    TPL,XPL,FXSPL,FXPL,RTPL,arrays storing the values of T,X,FXS,FX and
        RT respectively in the output points.
    DX, array containing the instantaneous introductions of units at the
        start point of each sub-region;
begin real DT,H,HMIN,E,T,TM,Z,XM,YM,R1,R2;
    integer I,J,K,L,M,N,I1,I2,MI,M1,M2,NI,NS,N1,NR,N2,NP,U,V;
    read N,DT,H,HMIN,E,MI,NI,NS,N1,NR N2,NP;
    if N> NP then I:=N else I:=NP;
    begin integer array MA[1:NI],MM[1:2*NI],NN[0:N1],N3,N4[0:N2],NM[0:N2,1:MI],
        N5,N6[0:NP];
        M:=0;
        for L:=1 step 1 until NI do begin read MA[L];M:=M+MA[L] end;
        M:=M+NI;
        punch(1);
        print ££l s1?NUMBER OF EQUATIONS, N =?,sameline,digits(3),N,
           ££l s1?A GUESS FOR THE STEP SIZE,H =?,sameline,scaled(4),H,
           ££l s1?LOWER BOUND FOR THE STEP SIZE,HMIN =?,sameline,HMIN,
           ££l s1?DESIRED BOUND FOR RELATIVE ACCURACY, E =?,sameline,E;
        begin array X,F,FXS,RT[1:N],TPL[0:M];
           comment CBS :; array XPL[1:M,1:I],FXSPL,RTPL[1:M,1:N],FX[1:N,1:MI],
              FXPL[1:M,1:N,1:MI],DX[1:NI,1:N];
           switch S:=L1,L2,L3,L4,L5,L6,L7,L8,L9;
           comment Read and print initial values;
           read T;TPL[0]:=T;
           for L:=1 step 1 until NI do
           for J:=1 step 1 until N do read DX[L,J];
           print ££l3s1?INSTANTANEOUS INTRODUCTION OF UNITS:?,££l2s9?T?;
           for I:=1 step 1 until N do print sameline,££s6?DX?,digits(2),I;
           TM:=T;
           for L:=1 step 1 until NI do
           begin print ££l??,sameline,scaled(4),TM;
              for J:=1 step 1 until N do print sameline,scaled(5),DX[L,J];
              TM:=TM+MA[L]*DT;
           end;
```

```
print ££13s9?T?;
for I:=1 step 1 until N do
print sameline,££s7?X?,digits(2),I;
for J:=1 step 1 until N do begin XPL[1,J]:=DX[1,J];X[J]:=0.0 end;
M2:=0;
comment Start integration of each sub-region with new initial
      values X[J];
for L:=1 step 1 until NI do
begin M1:=M2+1;M2:=M1+MA[L];MM[2*L-1]:=M1;MM[2*L]:=M2;
   TPL[M1]:=TPL[M1-1];print ££1??,sameline,scaled(4),T;
   for J:=1 step 1 until N do
   begin X[J]:=X[J]+DX[L,J];print sameline,scaled(5),X[J];
      if L>1 then XPL[M1,J]:=100.0*X[J]/XPL[1,J];
   end;
   comment Call FUNC for computation of the output variables
      at the start point of sub-region no. L;
   FUNC(N,MI,T,X,F,FXS FX,RT,L);
   for J:=1 step 1 until N do
   begin FXSPL[M1,J]:=FXS[J];RTPL[M1,J]:=RT[J];
      for K:=1 step 1 until MI do
      FXPL[M1,J,K]:=FX[J,K];
   end;
   comment TM=the value of T at the end point of sub-region no. L;
   TM:=T+MA[L]*DT+0.1;M:=M1;
   comment Call MERSON and FUNC repeatedly for computation of the
      output variables in each output point in sub-region no. L;
   for Z:=T+DT step DT until TM do
   begin MERSON(N,MI,T,Z,H,HMIN,E,X,F,FXS,FX,RT,L);
      FUNC(N,MI,T,X,F,FXS,FX,RT,L);
      M:=M+1;TPL[M]:=T;
      comment Print T and X in output point no. M. Store the same
            values for later plotting;
      Print ££1??,sameline,scaled(4),T;
      for J:=1 step 1 until N do
      begin print sameline,scaled(5),X[J];
         XPL[M,J]:=100.0*X[J]/XPL[1,J];
         FXSPL[M,J]:=FXS[J];RTPL[M,J]:=RT[J];
         for K:=1 step 1 until MI do
         FXPL[M,J,K]:=FX[J,K];
      end;
    end;
 end for L;
 comment Plot X[I],I=1,N, vs time;
 XM:=100.0;
 for J:=1 step 1 until N do
 for L:=2 step 1 until 2*NI do
 begin M1:=MM[L];if XPL[M1,J]>XM then XM:=XPL[M1,J];end;
 AXPLOT(TM,XM,R1,R2,I1,I2);
 movepen(350.0/R1,-40.0/R2);
print plotter(10,1),£Time(days)?;
 movepen(-50.0/R1,100.0/R2);
 print plotter(10,3),£Number of units in per cent of initialvalue?;
 movepen(-180.0/R1,500.0/R2);
 print plotter(10,1),£Initial?;movepen(-180.0/R1,480.0/R2);
 print plotter(10,1),£values:?;
```

```
for J:=1 step 1 until N do
begin movepen(-180.0/R1,(480.0-20.0*J)/R2);
   print plotter(10,1),£X?,digits(2),J,£=?,aligned(4,0),XPL[1,J];
end;
for J:=1 step 1 until N do
begin movepen(TPL[1],100.0);
   penlower;
   for I:=2 step 1 until M do
   drawline(TPL[I],XPL[I,J]);
   print plotter(10,1),£X?,digits(2),J;
end of plot X[I];
movepen(0,20*I2);
comment Plot X[I] vs X[J] for specified index pairs (I,J);
if NS=0 then goto L1;
for K:=1 step 1 until NS do
begin read U,V;
    for M1:=V do
    begin for M2:=U do
        begin XM:=YM:=100.0;
            for L:=2 step 1 until 2*NI do
            begin if XPL[MM[L],M1]>XM then XM:=XPL[MM[L],M1];
                if XPL[MM[L],M2]>YM then YM:=XPL[MM[L],M2];
            end;
            AXPLOT(XM,YM,R1,R2,I1,I2);
            movepen(100.0/R1,-40.0/R2);
            print plotter(10,1),
            £Number of units in per cent of initialvalue, X?,digits(2),M1;
            movepen(-50.0/R1,50.0/R2);
            print plotter(10,3),
            £Number of units in per cent of initialvalue, X?,digits(2),M2;
            movepen(-180.0/R1,500.0/R2);
            print plotter(10,1),£Initial?;movepen(-180.0/R1,480.0/R2);
            print plotter(10,1),£values:?;
            for J:=1,2 do
            begin movepen(-180.0/R1,(480.0-20.0*J)/R2);
               print plotter(10,1),£X?,digits(2),(if J=1 then M1 else M2),£=?,
                  aligned(4,0),XPL[1,(if J=1 then M1 else M2)];
            end;
            movepen(100,100);
            print plotter(10,1),£ T=?,aligned(2,0),TPL[1];
            movepen(100,100);
            penlower;
            for I:=2 step 1 until M do
            drawline(XPL[I,M1],XPL[I,M2]);
            movepen(XPL[M,M1]-60.0/R1,XPL[M,M2]);
            print plotter(10,1),£T=?,aligned(3,0),TPL[M];
            movepen(0,20*I2);
        end;
    end;
 end of plot X[I] vs X[J];
```

```
         comment Print FXS[I] vs time for specified indices I;
   L1:   punch(1);if N1=0 then goto L2;
         print ££13s9?T?;
         for I:=1 step 1 until N1 do
         begin read U;
            NN[I]:=U;print sameline,££s7?F?,digits(2),U;
         end;
         for J:=1 step 1 until M do
         begin print ££1??,sameline,scaled(4),TPL[J];
            for I:=1 step 1 until N1 do
            print sameline,scaled(5),FXSPL[J,NN[I]];
         end of print FXS[I];
         comment Print FX[I,J]/FXS[I] for specified indices I, and specified
               indices J for each I;
  L2:   if N2=0 then goto L3;
         print ££13s9?T?;
         for J:=1 step 1 until N2 do
         begin read U,M1;N3[J]:=U;N4[J]:=M1;
            for K:=1 step 1 until M1 do
            begin read V;
               print sameline,££s4?F?,digits(2),U,V;
               NM[J,K]:=V;
               for I:=1 step 1 until M do
               FXPL[I,U,V]:=100.0*FXPL[I,U,V]/FXSPL[I,U];
            end;
            print ££1s10??;
         end;
         for I:=1 step 1 until M do
         begin print ££1??,sameline,scaled(4),TPL[I];
            for J:=1 step 1 until N2 do
            begin M1:=N3[J];
               for K:=1 step 1 until N4[J] do
               begin M2:=NM[J,K];
                  print sameline,scaled(5),FXPL[I,M1,M2];
               end;
               print ££1s10??;
            end;
         end of print FX[I,J]/FXS[I];
         comment Print FXS[I]/FXS[J] for specified index pairs (I,J);
   L3:   if NP=0 then goto L4;
         print ££13s9?T?;
         for J:=1 step 1 until NP do
         begin read U,V;N5[J]:=U;N6[J]:=V;
            print sameline,££s??F?,digits(2),U,£/F?,V;
            for I:=1 step 1 until M do
            XPL[I,J]:=FXSPL[I,U]/FXSPL[I,V];
         end;
         for I:=1 step 1 until M do
         begin print ££1??,sameline,scaled(4),TPL[I];
            for J:=1 step 1 until NP do
            print sameline,scaled(5),XPL[I,J];
         end of print FXS[I]/FXS[J];
```

```
        comment Plot FXS[I] vs time;
L4:     if N1=0 then goto L5;
        XM:=100.0;
        for J:=1 step 1 until N1 do
        begin M1:=NN[J];
           for I:=2 step 1 until M do
           FXSPL[I,M1]:=100.0*FXSPL[I,M1]/FXSPL[1,M1];
           for L:=2 step 1 until 2*NI do
           begin M2:=MM[L];if FXSPL[M2,M1]>XM then XM:=FXSPL[M2,M1] end;
        end;
        AXPLOT(TM,XM,R1,R2,I1,I2);
        movepen(350.0/R1,-40.0/R2);
        print plotter(10,1),£Time(days)?;
        movepen(-50.0/R1,10.0/R2);
        print plotter(10,3),£Numb. of units lost per unit time, in per cent
              of init.val.?;
        movepen(-180.0/R1,500.0/R2);
        print plotter(10,1),£Initial?;movepen(-180.0/R1,480.0/R2);
        print plotter(10,1),£values:?;
        for J:=1 step 1 until N1 do
        begin movepen(-180.0/R1,(480.0-20.0*J)/R2);M1:=NN[J],
           print plotter(10,1),£F?,digits(2),M1,£=?,aligned(2,2),FXSPL[1,M1];
        end;
        for J:=1 step 1 until N1 do
        begin movepen(TPL[1],100.0);M1:=NN[J];penlower;
           for I:=2 step 1 until M do drawline(TPL[I],FXSPL[I,M1]);
           print plotter(10,1),£F?,digits(2),M1;
        end of plot FXS[I];
        movepen(0,20*I2);
        comment Plot FX[I,J]/FXS[I];
L5:     if N2=0 then goto L6;
        for J:=1 step 1 until N2 do
        begin U:=N3[J];M1:=N4[J];XM:=0.0;
           for K:=1 step 1 until M1 do
           begin V:=NM[J,K];
              for L:=1 step 1 until 2*NI do
              begin M2:=MM[L];H:=abs(FXPL[M2,U,V]);
                 if H>XM then XM:=H;
              end.
           end,
           AXPLOT(TM,XM,R1,R2,I1,I2);
           movepen(350.0/R1,-40.0/R2);
           print plotter(10,1),£Time(days)?;
           movepen(-50.0/R1,100.0/R2);
           print plotter(10,3),£F I J/F I ,in per cent?;
           for K:=1 step 1 until M1 do
           begin V:=NM[J,K];
              if FXPL[1,U,V]<0 then
              begin movepen(TPL[1],-FXPL[1,U,V]);penlower;
                 for I:=2 step 1 until M do drawline(TPL[I],-FXPL[I,U,V]);
                 print plotter(10,1),£-F?,digits(2),U,V,£/F?,U;
              end else
```

```
                 begin movepen(TPL[1],FXPL[1,U,V]);penlower;
                    for I:=2 step 1 until M do drawline(TPL[I],FXPL[I,U,V]);
                    print plotter(10,1),£+F?,digits(2),U,V,£/F?,U;
                 end;
              end;
              movepen(0,70*I2),
           end of plot FX[I,J]/FXS[I];
           comment Plot FXS[I]/FXS[J];
L6:        if NP=0 then goto L7;V:=0;N:=NP;
L9:        for J:=1 step 1 until N do
           begin XM:=0.0;
              for L:=1 step 1 until 2*NI do
              begin M1:=MM[L];
                 if XPL[M1,J]>XM then XM:=XPL[M1,J];
              end;
              E:=1.0;
              if XM>200.0 then
              begin K:=1;
                 for K:=K*10 while XM>200.0 do
                 begin M2:=K;XM:=XM/10.0 end;E:=1.0/M2;
              end;
              if XM<20.0 then
              begin K:=1;
                 for K:=K*10 while XM<20.0 do
                 begin M2:=K;XM:=XM*10.0 end;E:=M2;
              end;
              AXPLOT(TM,XM,R1,R2,I1,I2);
              movepen(350.0/R1,-40.0/R2),
              print plotter(10,1),£Time(days)?;
              movepen(-50.0/R1,200.0/R2);
              if V=0 then print plotter(10,3),freepoint(5),E,£*F I / F J?
              else print plotter(10,3),freepoint(5),E,£*R?,digits(2),N5[J];
              movepen(TPL[1],E*XPL[1,J]);penlower;
              for I:=2 step 1 until M do drawline(TPL[I],E*XPL[I,J]);
              if V=0 then print plotter(10,1),£F?,digits(2),N5[J],£ / F?,N6[J]
              else print plotter(10,1),£R?,digits(2),N5[J];
              movepen(0,70*I2);
           end of plot FXS[I]/FXS[J];
           if V=1 then goto L8;
           comment Plot RT[I] vs time for specified indices I;
L7:        if NR=0 then goto L8;V:=1;N:=NR;
           for J:=1 step 1 until NR do
           begin read U;N5[J]:=U;
              for I:=1 step 1 until M do XPL[I,J]:=RTPL[I,U];
           end;goto L9;
           comment end of plot RT[I];
L8:        N:=N;comment Dummy;
        end;
     end;
  end;
end;
end;
```